ซอฟต์แวร์
อาจหาญ สัตยารักษ์

SOFT MAIL

สนุกกับภาษาซี ด้วย CAI

ก้าวไปอีกขั้นกับกราฟิกโหมดในซีซึ่งไม่แตกต่างไปกว่าปาสคาล ง่ายต่อการศึกษาและเรียนรู้

ในการทำงานด้านกราฟิกนอกจากภาษาเบสิกและภาษาปาสคาลแล้ว ยังมีอีกภาษาหนึ่งที่มีความสามารถในด้านนี้ไม่ด้อยไปกว่ากันคือ ภาษาซี โดยเฉพาะรุ่นที่พัฒนาโดยบริษัทเดียวกันกับที่พัฒนาเทอร์โบปาสคาลในซอฟต์แวร์ที่มีชื่อว่า เทอร์โบซี (TURBO C) ซึ่งมีอยู่หลายรุ่น แต่การใช้คำสั่งด้านกราฟิกต่าง ๆ นั้นเป็นคำสั่งเดียวกันกับเทอร์โบปาสคาล โดยเฉพาะผู้ที่เคยเขียนกราฟิกด้วยภาษาปาสคาลมาก่อนจะคุ้นกับคำสั่งต่าง ๆ นี้เป็นอย่างดีอยู่แล้ว ข้อตกลงเบื้องต้นคือ ผู้อ่านต้องหาซอฟต์แวร์เทอร์โบซีนี้ตามร้านคอมพิวเตอร์มาใช้งานก่อนพร้อมกับจัดตั้งระบบให้ถูกต้องตามที่เทอร์โบซีกำหนดให้เรียบร้อย ผู้เขียนจะไม่กล่าวถึงในเรื่องการเรียกการเก็บไฟล์ การคอมพายล์ ฯลฯ ตลอดจนการใช้เป็นพิมพ์ต่าง ๆ โดยขออธิบายเฉพาะเรื่องการเขียนโปรแกรมเท่านั้น การเรียนรู้ตามบทความนี้ผู้เรียนไม่จำเป็นต้องรู้ภาษาซีอย่างลึกซึ้งก็ได้ ขอเพียงแต่ได้อ่านคำอธิบายและคีย์โปรแกรมหรือเรียนลัดโดยสั่งซื้อซอฟต์เมล์เลยก็ได้

การเข้าสู่กราฟิก

การใช้งานด้านกราฟิกบนเทอร์โบซีจะต้องอาศัยการเรียกใช้ไฟล์ด้านกราฟิก 2 ไฟล์ คือ graphics.h และ graphics.lib ซึ่งจะต้องอยู่ในพาธ (path) ที่กำหนดไว้ตอนจัดตั้งเทอร์โบซีและต้องใช้ไฟล์ *.bgi เพื่อจัดการกับจอภาพกราฟิกที่ตรงกับการ์ดที่ใช้อยู่ โดยไฟล์กลุ่มนี้จะอยู่ในพาธที่กำลังทำงานหรือในพาธอื่นก็ได้ แต่ต้องกำหนดคำสั่งให้ตรงกับพาธที่ไฟล์อยู่เท่านั้น หากต้องการจะใช้รูปแบบของตัวอักษรที่เปลี่ยนไปก็ต้องมีไฟล์ *.chr เข้ามาใช้งานร่วมด้วยโดยไฟล์กลุ่มนี้ต้องอยู่ในพาธที่กำลังติดต่ออยู่เท่านั้น การใช้งานไฟล์ graphics.h นั้นต้องมีการกำหนดหัวโปรแกรมด้วยคำสั่ง # include ตามที่แสดงในรูปที่ 1 ส่วนไฟล์อื่น ๆ นั้น เทอร์โบซีจะดำเนินการให้เองเมื่อมีการคอมพายล์หรือรันโปรแกรม คำสั่งที่เข้าสู่กราฟิกคือ

```
initgraph(int *graphdriver;
int *graphmode;
int pathofdriver);
```

```
#include <graphics.h>
#include <bios.h>
#include <math.h>

int  i,y,color,crttype,cx,cy,rx,ry,angst,angend;

void opengraph()

  {
   int typecrt, grdriver, grmode;
       grmode=1; grdriver=9;  crttype=1;
       typecrt=biosequip();
        if ((typecrt & 48)== 48)      {
            crttype=0;
            grmode=0;  grdriver=7;    }
       initgraph(&grdriver,&grmode,"");
  }

void set_color(int colors)
 {
   setcolor(colors); color=colors;
   if (crttype == 0)
        { setcolor(1); color=1; }
 }
```

รูปที่ 1 ชุดคำสั่งในการเข้าสู่กราฟิกและการกำหนดสี

graphics_driver constant	nummeric value
DETECT	0
CGA	1
MCGA	2
EGA	3
EGA64	4
EGAMONO	5
RESERVED	6
HERCMONO	7
ATT400	8
VGA	9
PC3700	10

ตารางที่ 1 แสดงค่าของตัวแปร graphdriver ตามชนิดของจอภาพ

graphics_mode		value	column × row	palette	page
CGA	CGACO	0	320×200	C0	1
	CGAC1	1	320×200	C1	1
	CGAC2	2	320×200	C2	1
	CGAC3	3	320×200	C3	1
	CGAHI	4	320×200	2 color	1
EGA	EGALO	0	640×200	16 color	4
	EGAHI	1	640×350	16 color	2
EGAMONO	EGAMONOHI	3	640×350	2 color	1(2)*
HERCMONO	HERCMONOHI	0	640×348	2 color	2
VGA	VGALO	0	640×200	16 color	2
	VGAMED	1	640×350	16 color	2
	VGAHI	2	640×480	16 color	1

ตารางที่ 2 แสดงขนาดของกราฟิกตามตัวแปร graphmode (เฉพาะจอภาพ)

*graphdriver คือตัวแปรที่กำหนดชนิดของจอภาพมีค่า 0-10

*graphmode คือตัวแปรกำหนดขนาดของกราฟิก มีค่าตามชนิดของจอภาพ

*pathofdriver คือการกำหนดพาธที่ไฟล์ *.bgi อยู่ เช่น "" คือ พาธที่กำลังติดต่ออยู่ หรือ "c:\driver" คือ อยู่ในไดรฟ์ c, path ชื่อไดรฟ์เป็นต้น แต่ส่วนมากจะกำหนดเป็น "" *graphdriver หรือ *graphmode เป็นชื่อตัวแปรที่สามารถกำหนดเป็นอย่างอื่นได้ โดยมีรายละเอียดในการกำหนดค่าตัวแปรทั้งสองดังตารางที่ 1 และตารางที่ 2 จากตารางที่ 1 ถ้ากำหนดให้เท่ากับ DETECT เป็นการตรวจชนิดของจอภาพโดยอัตโนมัติ แต่มีข้อเสียคือจะได้ขนาดของกราฟิกตามที่กำหนดมาไว้ให้เท่านั้น ซึ่งอาจไม่ตรงกับกราฟิกที่เราต้องการก็ได้ เช่น ถ้าใช้กับจอวีจีเอจะได้กราฟิกขนาด 640×480 จุดเป็นต้น ดังนั้นถ้าเราต้องการจะใช้กราฟิกขนาดอื่นจึงควรกำหนดค่าเองจะดีกว่า คำสั่งในการตรวจชนิดของจอภาพคือ biosequip( ); ซึ่งจะต้องทำการ #include <bios.h> มาใช้งานด้วย ดังนั้นคำสั่งที่สมบูรณ์ในการเข้าสู่กราฟิกจึงเป็นดังรูปที่ 1 ซึ่งในบทความครั้งต่อไปจะไม่แสดงชุดคำสั่งนี้อีก แต่ขอให้เรียกเข้ามาใช้ด้วยทุกครั้งหากจะเก็บเป็นไฟล์ที่เรียกเข้ามาด้วยคำสั่ง #include ก็ต้องเก็บไฟล์ไว้ในพาธของการ include ด้วยเช่นกัน

รายละเอียดของคำสั่ง opengraph( ) ( ) มีดังนี้

1. กำหนดตัวแปร crttype เป็นตัวแปรโกบอล (กำหนดไว้ที่ต้นโปรแกรม) ส่วนตัวแปร grdriver, grmode และ typecrt เป็นตัวแปรโลคอล

2. เริ่มด้วยการกำหนดตัวแปร grmode = 1; grdriver = 9; crttype = 1; เพื่อเรียกใช้จอสีวีจีเอในกราฟิกขนาด 640×350 จุด 16 สี 2 หน้ากราฟิก

3. ตรวจชนิดของจอภาพจากคำสั่ง typecrt = biosequip( ); โดยตรวจที่ค่าของบิตที่ 4 และ 5 โดยมีค่าดังตารางที่ 3

ค่าที่อ่านได้ (ฐานสอง)	ค่าเลขฐานสิบ	ลักษณะการ์ดที่ใช้
00	00	Unused
01	16	40×25 BW with color card
10	32	80×25 BW with color card
11	48	80×25 BW with mono card

ตารางที่ 3 แสดงการตรวจสอบค่าของบิตที่ 4 และ 5

ตามคำสั่งได้ตรวจที่จอโมโนโครม คือ typecrt & 48 == 48 ถ้าใช่ก็กำหนดตัวแปรใหม่ตามชนิดของจอโมโนโครม ส่วนตัวแปร crttpye กำหนดเพื่อนำไปใช้ในชุดคำสั่ง set__color( ); หรือชุดคำสั่งอื่น ๆ

4. เมื่อได้ค่าของตัวแปรตามชนิดของจอภาพแล้วก็เรียกใช้คำสั่ง initgraph ตามรูปแบบที่กำหนดในบรรทัดสุดท้ายของชุดคำสั่ง

5. เรียกใช้ชุดคำสั่งนี้ด้วยคำสั่ง opengraph( ); เมื่อเข้าสู่กราฟิกของจอสีและจอโมโนโครมโดยจะเป็นกราฟิกในขนาดที่ใกล้เคียงกันคือ 640×350 จุดและ 720×348 จุดตามลำดับ

ดังนั้นถ้าต้องการให้โปรแกรมรันได้อย่างสมบูรณ์ทั้งจอสีและจอโมโนโครมก็ควรกำหนดด้วยกราฟิกนี้แต่เขียนภาพในขอบเขตที่เหมาะสมกับทั้งสองจอภาพ คือ 640×348 จุด ส่วนคำสั่ง set__color( ); นั้นมีไว้เพื่อกำหนดสีของภาพ เพราะว่าจอสีสามารถกำหนดตัวเลขได้จาก 0-15 ส่วนจอโมโนโครมกำหนดได้แค่ 0-1 เท่านั้น ดังนั้นถ้าเขียนโปรแกรมบนจอสีด้วยการกำหนดค่าสีเท่ากับหมายเลข 2 เมื่อนำมารันบนจอโมโนโครมจะได้ค่าสีเป็นหมายเลข 0 แทนซึ่งมีผลให้ไม่เกิดภาพบนจอ ดังนั้นชุดคำสั่ง set__color จึงมาแทนชุดคำสั่งที่มีอยู่แล้วชื่อ setcolor โดยถ้าเป็นจอสีก็ลงสีตามที่กำหนดค่ามาให้ แต่ถ้าเป็นจอโมโนโครม (ตัวแปร crttype = 0) ก็จะลงสีหมายเลข 1 ส่วนการกำหนดสีหมายเลข 0 ควรใช้ชุดคำสั่ง setcolor ของเดิมเพราะให้ผลในการลบภาพแบบเดียวกัน รูปแบบของคำสั่ง setcolor คือ setcolor(int color); โดยค่าของตัวแปร color มีขอบเขตตามชนิดของจอภาพ ส่วนหมายเลขใดเป็นสีใดก็ขอให้ลองใช้คำสั่งดูเอง (ทำบนจอสี) การออกจากกราฟิกทำได้โดยการใช้คำสั่ง closegraph( );

คำสั่งด้านกราฟิก

สำหรับตอนนี้เราจะเรียนรู้คำสั่งด้านกราฟิกบางคำสั่ง หากมีการทำงานในส่วนที่เกินขอบเขตของคำสั่งที่เราเรียนรู้ก็จะเขียนชุดคำสั่งขึ้นมาช่วยงานก่อน โดยชุดคำสั่งนี้อาจมีคำสั่งอื่น แต่การเขียนจะช่วยให้เรามองเห็นโครงสร้างของคำสั่งในงานกราฟิกได้ดียิ่งขึ้น ดังนั้นหากเราเรียนรู้คำสั่งมากขึ้น ผู้อ่านควรนำมาปรับปรุงโปรแกรมที่เคยเรียนรู้มาก่อนหน้าให้มีความกระชับมากขึ้นด้วย

คำสั่งด้านกราฟิกอื่น ๆ

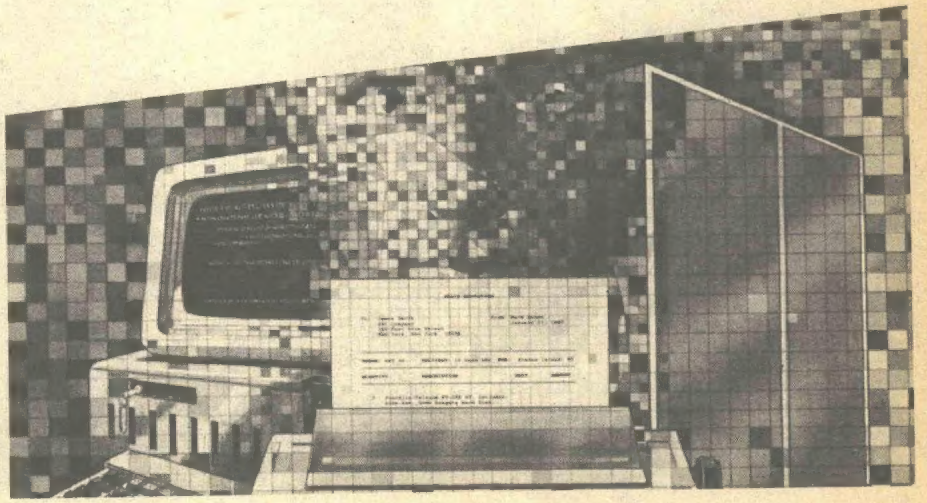
1. การลงจุด มีรูปแบบคือ

```
putpixel (int x, int y, int pixelcolor);
```

x, y คือคู่ลำดับบนจอภาพที่ต้องการลงจุด (คู่ลำดับ 0, 0 อยู่ที่มุมบนซ้ายมุมล่างขวาเป็นค่าสูงสุดของแกน x และแกน y)

pixelcolor คือสีของจุดที่ลง มีค่าตามชนิดของจอภาพที่ใช้

เมื่อต้องการจะตรวจว่ามีการลงจุดหรือไม่ก็ใช้คำสั่ง getpixel(int x, int y); ซึ่งทำให้ทราบได้ว่ามีการลงจุด



หรือไม่ลงจุดเพื่อนำไปใช้ประโยชน์อื่นได้เช่น ใช้ในการเขียนโปรแกรมพิมพ์ภาพจากจอกราฟิกลงบนกระดาษ (ดูตัวอย่างได้ใน "ของฝาก" ฉบับนี้)

2. การลากเส้น มีรูปแบบคำสั่งดังนี้

```
line (int x1, int y1, int x2, int y2);
```

x1, y1 คือคู่ลำดับเริ่มต้นในการลากเส้นตรง

x2, y2 คือคู่ลำดับสุดท้ายที่จะลากเส้นตรง

สีของเส้นต้องกำหนดล่วงหน้าก่อนจะมีการลากเส้น การลากเส้นแบบนี้จะเป็นการแสดงอย่างรวดเร็วถ้าต้องการจะแสดงทีละจุดต้องเขียนคำสั่งขึ้นใช้เองเช่นกัน

3. การวาดรูปสี่เหลี่ยม มีรูปแบบดังนี้

```
bar (int x1, int y1, int x2, int y2);
```

x1, y1 คือคู่ลำดับของมุมบนซ้ายของรูปสี่เหลี่ยม

x2, y2 คือคู่ลำดับของมุมล่างขวาของรูปสี่เหลี่ยม

การกำหนดคู่ลำดับของมุมสลับที่กันก็จะมีการตรวจสอบให้ส่วนการกำหนดสีจะกำหนดได้สองอย่าง คือ สีพื้นและสีขอบ โดยสีพื้นกำหนดในคำสั่ง setfillstyle ส่วนสีขอบกำหนดด้วยคำสั่ง setcolor

4. การกำหนดรูปแบบพื้น ใช้คู่กับคำสั่งที่มีการวาดพื้นของจอภาพมีรูปแบบดังนี้

```
setfillstyle (int pattern, int color);
```

pattern กำหนดได้ 11 แบบค่าตัวเลข 0-10 มีรูปแบบอย่างไรบ้างก็ขอให้ลองเขียนโปรแกรมรันดู

color คือสีของพื้นที่ทำการระบายสีมีค่าตามชนิดของจอภาพ

5. การระบายพื้น ใช้เมื่อต้องการระบายพื้นตามรูปแบบที่ได้กำหนดด้วยคำสั่ง setfillstyle ลงในรูปปิด (รูปที่ไม่ปิดการระบายจะล้นออกนอกรูป) มีรูปแบบดังนี้

```
floodfill (int x, int y, int border);
```

x, y คือคู่ลำดับที่กำหนดการลงจุด ตำแหน่งนี้จะต้องไม่ใช่ตำแหน่งที่ลงสีค่าเดียวกับสีของ border

border คือสีของขอบรูปปิดที่จะระบายจะต้องเป็นสีเดียวกันเท่านั้น (กรณีที่ใช้จอสีต้องระวังเรื่องสีนี้ให้มาก ๆ) หากรูปปิดมีสีคนละค่ากับตัวแปร border ก็จะระบายล้นออกนอกรูปเช่นกัน คำสั่งนี้เมื่อใช้กับจอสีจึงต้องตรวจเรื่องสีให้รอบคอบ

6. การแสดงข้อความบนจอกราฟิก มีรูปแบบดังนี้

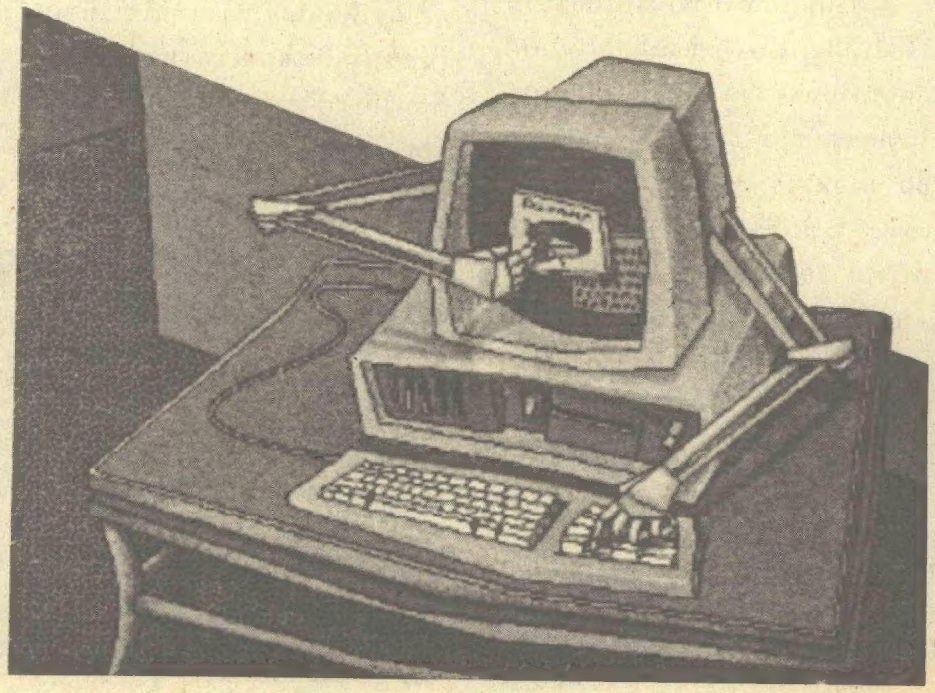
```
outtextxy (int x, int y, char far *textstring);
```

x, y คือ คู่ลำดับเริ่มต้นที่จะนำข้อความไปแสดง

textstring คือ ข้อความที่ต้องการแสดง

เครื่องฉายภาพนิ่ง

คราวนี้ก็มาถึงการนำคำสั่งที่เราได้เรียนรู้มาเขียนโปรแกรม ซึ่งในตอนนี้จะเป็นเรื่องเครื่องฉายภาพนิ่ง ซึ่งเป็นเนื้อหาวิชาฟิสิกส์ที่เรียนในระดับชั้น ม.4 (หลักสูตรใหม่) ตามรูปในหนังสือเรียนหน้า 74 เริ่มจากการกำหนดตัวแปรที่นำไปใช้งานก่อน มีตัวแปรตามที่กำหนดไว้ในตารางที่ 1 บรรทัดที่ 4 ซึ่งเป็นตัวแปรโกบอล คือจะนำไปใช้ในชุดคำสั่งใด ๆ ก็ได้ การวาดรูปกระจกเว้า เลนส์รวมแสงและเลนส์ฉายภาพ จะต้องเขียนส่วนโค้งซึ่งเรายังไม่ได้เรียนรู้คำสั่งนี้ โดยต้องเขียนชุดคำสั่งขึ้นมาใช้งาน คือชุดคำสั่ง drawcircle ดังที่ปรากฏในรูปที่ 2 พร้อมกับมีชุดคำสั่ง Slide Project เป็นการวาดภาพของทัศน-



```
void drawcircle()
{
 int x,y,xx,yy;
 float ang;
  for (ang =angst;  ang <=angend; ang=ang+2)
    {
      x=cx+(int)(rx*cos(ang/180*22/7));
      y=cy-(int)(ry*sin(ang/180*22/7));
      if (ang==angst) { putpixel(x,y,color); }
        else line(xx,yy,x,y);
      xx=x; yy=y;
    };
}

void SlideProjector()
  {
       outtextxy(200,60,"Slide Projector");
      /* draw concave mirror  */
         cx=70; cy=170; rx=40;  ry=50;
         angst=110; angend=250; drawcircle();
      /* draw condenser lens    */
         cx=180;  rx=30; ry=60;  setfillstyle(9,color);
         angst=290; angend=430;  drawcircle();
         line(190,115,190,225);  floodfill(195,170,color);
         cx=280;  angst=110; angend=250;  drawcircle();
         line(270,115,270,225); floodfill(268,170,color);
      /* draw projecting lens    */
         cx=380;  rx=5; ry=25;
         angst=0; angend=360; drawcircle();
         floodfill(380,170,color);
      outtextxy(5,220,"concave mirror");
      outtextxy(174,234,"condenser lens");
      outtextxy(335,208,"projecting lens");
  }

void Lamp()
 {
 /* draw lamp */
  setfillstyle(9,color);      bar(93,176,108,190);
  cx=100; cy=170; rx=12; ry=12;
  angst=0; angend=360; drawcircle();
  line(95,180,95,165);        line(95,165,100,173);
  line(100,173,105,165);      line(105,165,105,180);
  outtextxy(95,148,"lamp");  getch();
  /* draw object */
   setfillstyle(1,color);        bar(340,150,342,185);
   line(342,185,346,185);        line(346,185,341,190);
   line(341,190,336,185);        line(336,185,340,185);
   outtextxy(325,140,"object");  getch();
```

รูปที่ 2 ชุดคำสั่งในการวาดวงกลมและส่วนประกอบของเครื่องฉายภาพนิ่ง

อุปกรณ์ทั้งสามลงจอภาพซึ่งจะต้องอยู่ในแนวเส้นตรงเดียวกันด้วย ขอให้ศึกษาชุดคำสั่งทั้งสองนี้ด้วยตนเองก่อน

รายละเอียดของชุดคำสั่ง drawcircle ( ) มีดังนี้

1. กำหนดตัวแปร x, y, xx, yy เป็นชนิด integer และตัวแปร ang เป็นเลขจำนวนจริง

2. วนรอบทำงานในขอบเขตค่า ang ตั้งแต่ angst ไปจนถึง angend (ค่าตัวแปร angst, angend กำหนดไว้ก่อนเรียกใช้ชุดคำสั่งนี้ โดยกำหนดมุมในหน่วยองศา) การวนรอบนี้จะเพิ่มค่า ang ครั้งละ 2 (จากคำสั่ง ang = ang +2) หากต้องการให้วาดเร็วขึ้นก็ให้เปลี่ยนเลข 2 เป็นค่าที่มากขึ้น โดยในช่วงวนรอบได้มีการทำงานดังนี้

2.1 คำนวณหาค่าของ x และ y โดยอาศัยรัศมีของแกน (rx, ry) ที่กำหนดไว้ก่อนเรียกใช้ชุดคำสั่งนี้ การคำนวณนี้ใช้มุมในหน่วยเรเดียนจึงต้องเปลี่ยนค่ามุมองศาเดิมให้เป็นหน่วยเรเดียนเสียก่อน โดยถือหลักว่ามุม 180 องศามีค่าเท่ากับ π เรเดียน

2.2 ถ้าค่า ang = angst ให้ทำการลงจุด ณ ตำแหน่ง x, y ด้วยคำสั่ง putpixel

2.3 ถ้าค่า and > angst ให้ทำการลากเส้นระหว่างจุด xx, yy ไปยังจุด x, y ด้วยคำสั่ง line

2.4 กำหนดค่า xx = x; yy = y เพื่อให้ข้อ 2.3 ลากเส้นจากจุดที่เคยลงไปยังจุดใหม่ที่คำนวณได้ ซึ่งจะมีผลให้ได้ส่วนของวงกลมที่ต่อเนื่องกัน

ชุดคำสั่ง SlideProject( ) มีดังนี้

1. วาดรูปกระจกเงาเว้าด้วยมุมตั้งแต่ 110-250 องศา รัศมีแกน x = 40 และรัศมีแกน y = 50 โดยมีจุดศูนย์กลางอยู่ที่คู่ลำดับ 70, 170 โดยเรียกใช้ชุดคำสั่ง drawcircle( )

2. วาดรูปเลนส์รวมแสง (เลนส์นูนด้านหนึ่งเรียบมี 2 เลนส์วางประกอบกัน) ด้วยมุมตั้งแต่ 290-430 องศา ด้วยรัศมีแกน x = 30 และรัศมีแกน y = 60 มีจุดศูนย์กลางอยู่ที่คู่ลำดับ 180, 170 รูปหนึ่ง และอีกรูปหนึ่งวาดด้วยมุมตั้งแต่ 110-250 ณ จุดศูนย์กลางที่ 280, 170 ใช้คำสั่ง drawcircle( ); เช่นกัน ทั้งสองรูปได้วาดให้เป็นรูปปิดด้วยคำสั่ง line และระบายพื้นด้วยคำสั่ง floodfill โดยกำหนดพื้นรูปแบบที่ 9 ตามคำสั่ง setfillstyle (9, color)

3. วาดรูปเลนส์ฉายภาพซึ่งเป็นเลนส์นูนสองหน้า โดยกำหนดมุมตั้งแต่ 0-360 องศา รัศมีเป็น 5, 25 ตามลำดับใน

แกน x และ y โดยมีจุดศูนย์กลางที่คู่ลำดับ 380, 170 พร้อมกับทำการระบายพื้น

4. แสดงข้อความลงในตำแหน่งต่าง ๆ ด้วยคำสั่ง outtextxy

มาถึงตรงนี้คิดว่าคงพอจะสามารถทำความเข้าใจกับโปรแกรมได้บ้างแล้ว จึงไม่ขออธิบายชุดคำสั่ง lamp( ) ซึ่งจะวาดรูปของหลอดไฟและวัตถุ (หรือแผ่นสไลด์) คำสั่ง getch( ); คือคำสั่งสำหรับรอการกดแป้น

การแสดงรังสีของแสง

เพื่อให้เห็นการเกิดภาพของสไลด์ไปปรากฏบนจอเราทำได้โดยการลากรังสีของแสง 2 ชุด เพื่อแสดงให้เห็นการทำงานของทัศนอุปกรณ์ต่าง ๆ ดังนี้

1. แสงจากหลอดไฟ (ตำแหน่งของหลอดไฟอยู่ที่จุดโฟกัสของกระจกเงาเว้าและจุดโฟกัสของเลนส์รวมแสง) ที่ไปทางด้านหลังจะกระทบกับกระจกเงาเว้าแล้วสะท้อนออกมาในแนวขนานกับเส้นแกนมุขสำคัญผ่านเลนส์รวมแสงและถูกเบนเข้ารวมกันผ่านวัตถุไปพบกับเลนส์ฉายภาพแล้วรังสีนี้จะเดินทางไปกระทบจอ

2. แสงที่ไปทางด้านหน้าจะผ่านเลนส์รวมแสงแล้วหักเหในแนวขนานกับเส้นแกนมุขสำคัญไปกระทบวัตถุแล้วหักเหผ่านเลนส์ฉายภาพไปตัดกับรังสีในข้อ 1 ทำให้เกิดภาพจริงปรากฏบนจอรับภาพ

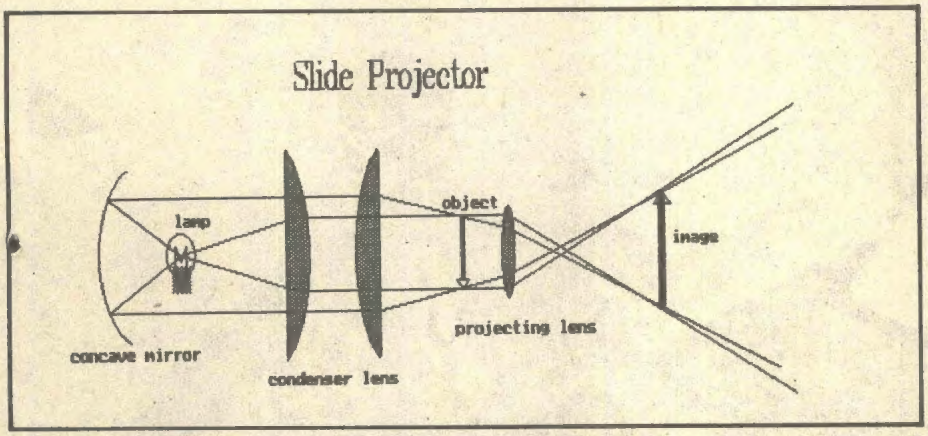
เราจึงเขียนโปรแกรมเพื่อให้แสดงรังสีทั้งสองแบบนี้ตามชุดคำสั่งในรูปที่ 3 ในชุดคำสั่ง drawray1 ( ) และ drawray2( ) ทำให้เกิดภาพด้วยชุดคำสั่ง image ( ) รังสีที่เขียนเป็นตัวอย่างนี้อาจไม่ถูกต้องตรงกับความจริงนักแต่ก็แสดง

```
void DrawRay1()
 {
     set_color(13);
   for (i=0; i<62; i++)
     {
       putpixel(100-i,170+i/2,color);
       putpixel(100-i,170-i/2,color);  delay(10);
     }
   for (i=0; i<233; i++)
     {
       putpixel(38+i,201,color);
       putpixel(38+i,139,color);       delay(10);
     }
   for (i=0; i<110; i++)
     {
       putpixel(271+i,201-i/6,color);
       putpixel(271+i,139+i/6,color);  delay(10);
     }
   for (i=0; i<230; i++)
     {
       putpixel(382+i,182-i/3,color);
       putpixel(382+i,158+i/3,color);  delay(10);
     }
 }

void DrawRay2()
 {
     set_color(3);
   for (i=0; i<98; i++)
     {
       putpixel(100+i,170-i/5,color);
       putpixel(100+i,170+i/5,color);  delay(10);
     }
   for (i=0; i<185; i++)
     {
       putpixel(198+i,190,color);
       putpixel(198+i,150,color);      delay(10);
     }
   for (i=0; i<245; i++)
     {
       putpixel(380+i,190-i/5*2,color);
       putpixel(380+i,150+i/5*2,color); delay(10);
     }
 }

void image()
 {
   set_color(14);
   setfillstyle(1,color);   bar(509,140,513,200);
   line(510,140,505,145); line(505,145,510,145);
   line(512,145,519,145); line(519,145,512,140);
   outtextxy(520,160,"image");
 }
```

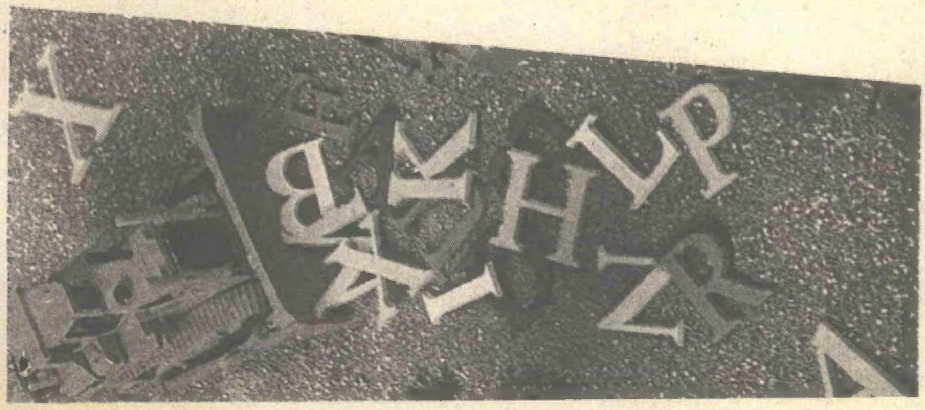
รูปที่ 3 ชุดคำสั่งแสดงการเดินทางของรังสีของแสงและการวาดภาพที่ได้

```
/**********  SlideProjector (TURBO C) by arthan sattayarak ************/

main()
 {
   opengraph();
     set_color(14);
     SlideProjector();
     Lamp();
     DrawRay1();
     DrawRay2();
     image();
     getch();
   closegraph();
 };
```

รูปที่ 4 โปรแกรมหลักที่นำชุดคำสั่งต่าง ๆ มาแสดงเป็นเครื่องฉายภาพนิ่ง



รูปที่ 5 ภาพที่พิมพ์จากจอภาพกราฟิกของโปรแกรมเครื่องฉายภาพนิ่ง

เป็นแนวทางให้เข้าใจถึงการเกิดภาพของวัตถุว่ามีลักษณะอย่างไร ซึ่งจะเห็นว่าได้ภาพจริงหัวกลับขนาดโตกว่าวัตถุ ขอให้ศึกษาการใช้คำสั่งในการลงจุดด้วยคำสั่ง putpixel ซึ่งมีผลให้แสดงการเคลื่อนที่ของรังสีของแสงปรากฏอย่างช้าด้วยคำสั่ง delay(10) หากต้องการจะให้ช้าหรือเร็วกว่านี้ให้เปลี่ยนตัวเลขในคำสั่งนี้ได้ สามารถให้ศึกษาคำสั่งต่าง ๆ ได้ในรูปที่ 3 ก่อน ซึ่งเป็นการวนรอบด้วยคำสั่ง for

เมื่อเราได้เขียนโปรแกรมด้านการทำงานอื่น ๆ จนครบแล้ว ต้องนำมารวมกันเพื่อให้เกิดการสร้างภาพปรากฏบนจอ โดยการเขียนคำสั่งในโปรแกรมหลักดังรูปที่ 4 โดยเริ่มต้นการกำหนดสีเป็นสีลำดับที่ 14 (สีเหลือง) ด้วยคำสั่ง set_color(14); ซึ่งจะได้สีหมายเลข 1 เมื่อรันในจอโมโนโครมตามที่ได้กำหนดการตรวจสอบค่าตัวแปรไว้แล้วและจบด้วยการออกจากกราฟิก (closegraph) เราจะได้ผลบนจอภาพหลังการเรียกใช้ชุดคำสั่ง image( ); ดังรูปที่ 5 ซึ่งรูปนี้ผู้เขียนได้พิมพ์จากหน้าจอภาพด้วยโปรแกรมพิมพ์ภาพกราฟิกด้วยภาษาซี โดยมีรายละเอียดอยู่ในคอลัมน์ของฝากฉบับนี้ด้วยแล้ว ให้ไปศึกษาการทำงานของโปรแกรมนั้นอีกที โดยเรียกเข้ามาในโปรแกรมแล้วเรียกใช้ด้วยคำสั่ง hardcopy ที่ต้องกำหนดคอลัมน์สำหรับการกั้นซ้ายของเครื่องพิมพ์ไปด้วย เช่น hardcopy(5) เป็นต้น